Foundations of Multivariate Functional Approximation for Scientific Data

Tom Peterka* Youssef S. G. Nashed† Iulian Grindeanu‡ Vijay S. Mahadevan§ Raine Yeh¶
Xavier Tricoche‖

ABSTRACT

The size and complexity of scientific data are rapidly outpacing our ability to manage, analyze, visualize, and draw conclusions from the results. One way to streamline scientific data processing is to transform the data model to one that is not only more efficient but that facilitates analytical reasoning. To this end, we are investigating modeling discrete scientific data by a functional basis representation based on a tensor product of nonuniform rational B-spline functions (NURBS). The functional model is an approximation that is more efficient to communicate, store, and analyze than the original form, while having geometric and analytic properties making it useful for further analysis without translating back to the discrete form. This paper presents four main contributions. The first is modeling in high dimensions, not just curves or surfaces but volumes and hypervolumes. The second is an adaptive algorithm that adds knots and control points until the maximum error is below a threshold set by the user. The adaptive algorithm also works in high dimensions. The third is precise evaluation of values and high-order derivatives anywhere in the domain, again in high-dimensional space. The fourth contribution is parallelization on high-performance supercomputers. We evaluate our model using both synthetic and actual scientific datasets.

1 INTRODUCTION

Scientific discovery through high-performance computing (HPC) depends on managing, analyzing, and visualizing data. While the HPC community's capability to generate more data continues to grow, the ability to store, transform, and ultimately to draw scientific conclusions from data is lagging. Rethinking the way that scientific data are represented is one way to break out of this spiral.

This paper presents initial findings in an ongoing investigation in redefining discrete scientific data by an alternative representation. We call *Multivariate Functional Approximation (MFA)* the approximation of raw discrete data with a hypervolume of piecewise-continuous functions. The MFA model can represent numerous simulations, experiments, analyses, and visualizations. Compared with existing discrete data models, the MFA model is designed to enable most spatiotemporal analyses without reverting back to the original discrete form, usually while occupying less storage space.

In this paper we provide the theory and implementation to model the MFA. We demonstrate an algorithm for adaptively adding data to the MFA until a user-set error threshold is achieved. We use the MFA to compute high-order values and derivatives at any point in the domain, not limited to the input point locations. We also demonstrate that our data model and algorithm are easily parallelized on HPC architectures.

*Argonne National Laboratory, tpeterka@mcs.anl.gov
†Argonne National Laboratory, ynashed@anl.gov
‡Argonne National Laboratory, iulian@mcs.anl.gov
§Argonne National Laboratory, mahadevan@anl.gov
¶Purdue University, raine.yeh@gmail.com
‖Purdue University, xmt@purdue.edu

What makes the MFA unique is that the transformed data retain geometric properties: spatial and temporal contiguity, derivatives and statistical distributions are all preserved. The goal of analyzing scientific data—which are inherently spatiotemporal—is usually to understand the relationship of science variables to their position in space and time. This is why once data are modeled in the MFA, many data analyses and visualizations are possible directly from it.

The maximum error in the MFA approximation is selectable by the user. Given the maximum error, our modeling algorithm is automatic, adaptively adding control points as needed in areas of high error. In this paper we demonstrate the following features of our model:

- Fitting multivariate spatiotemporal field geometry and science variable data with a separable dimension algorithm that lowers computation complexity.
- Progressively improving the quality of the approximation by locally adding more detail in spatial regions of rapid change (e.g., edges, turbulence, discontinuities).
- Building a parallel data model based on the MFA that scales to HPC architectures.
- Evaluating modeling and usage in both synthetic and actual scientific datasets.

Section 2 summarizes related work in lossy compression, functional data analysis, low-rank approximations, and applications of functional data representations. Section 3 is a primer on nonuniform rational B-spline (NURBS) tensor products. Section 4 explains how we extended these concepts to high dimensions and adaptively add control data where the error needs to be reduced. Section 5 demonstrates two key uses of the MFA: evaluating points anywhere in the domain and evaluating derivatives up to the degree of the model. Section 6 evaluates our model on synthetic and actual test data. Section 7 summarizes our work and outlines the next research topics we intend to pursue.

2 RELATED WORK

Multivariate functional approximation borrows ideas from several fields: compression, statistics, modeling, visualization, and analysis. Piecewise functional approximations replace discrete data points with linear combinations of basis functions and a small number of reference points called *control points*. Statisticians call this method *functional data analysis* [45, 16], with low-dimensional serial implementations available in popular statistics packages [44].

A key decision in functional data analysis is the choice of basis function family. Fourier [5], wavelet [21, 25], and geometric (i.e., spline) [12] bases are commonly used; recently Austin et al. [1] proposed the Tucker decomposition as a low-rank alternative. Majdisova and Skala proposed radial basis functions for particle data [32]. We use geometric (NURBS) bases for the MFA because they mirror the space-time properties in the original data and retain these geometric properties in analytics and visualization. Computing geometric bases requires minimal memory overhead, and the parallel decomposition of the original space-time domain is identical in the geometric functional domain, an important consideration for minimizing data movement at scale.

Compression of floating-point data is one way to reduce data size. Lindstrom and Isenburg [31] demonstrated lossless floating-point compression based on predictive coding using the Lorenzo predictor [24]. Laney et al. [28] and Lindstrom [30] demonstrated that valid

scientific conclusions can be drawn even from lossy-compressed data as well. To date, spline fitting methods in compression algorithms were limited to 1-d curves or 2-d surfaces. For example, piecewise curve fitting has been used to compress time-series and other 1-dimensional data [26], with higher dimensions being linearized in a single dimension using space-filling curves. Di and Cappello also perform curve fitting as the first step in their multistage compressor [14]. Chen et al. modeled flow field pathlines using quadratic Bezier curves [6]. Unlike compressors, the MFA retains key geometric properties making it directly usable later. Also, one could in theory further compress (lossy or lossless) the MFA using any of the above methods, assuming the total error bounds of the combination were understood.

Researchers demonstrated that NURBS can faithfully represent scientific data in up to three dimensions. 2D triangular surfaces and 3D tetrahedral meshes can be converted into bivariate and trivariate NURBS. Our MFA can be considered as a multivariate extension of trivariate NURBS to any number of dimensions. Martin and Cohen first developed the data model and framework in 2001 [34], and Martin et al. described how to parameterize triangular and tetrahedral data as tensor products in 2008 [33]. They demonstrated modeling of exterior surfaces and interior volumes of medical datasets.

Geometric functional representations can replace data models used today in visualization algorithms. Martin and Cohen derived isosurfacing, slicing, and ray tracing algorithms directly from NURBS models [34]. Raviv and Elber [46] showed direct rendering from trivariate B-splines for isosurface extraction, planar slicing, and volume rendering. Park and Lee [38] visualized trivariate NURBS for flow data. Hua et al. [22] modeled three-dimensional solids using simplex splines and showed that visualization algorithms such as isosurfacing and volume rendering can operate on the same data model.

Functional data models are used in mechanical and fluid engineering simulations. One example is isogeometric analysis (IGA) [23] that uses NURBS models for mechanical simulations. Recently, a parallel IGA toolkit [9] built on PETSc [2] was developed for solving high-order partial differential equations over NURBS basis domains. Spectral methods are another example of a discretization that uses basis functions to evaluate data: the Nek5000 Navier-Stokes solver [13] is based on a weighted sum of basis functions defined over a coarse set of control points.

NURBS modeling is a rich and mature topic in computer-aided geometric design (CAGD) literature. Lin et al. [29] summarized numerous approaches to solving for control point positions, weights, and optionally knots given a set of input points (called reverse engineering). These methods are limited to 3-d geometry of exterior surfaces. If the knots and spline parameter values are assigned in advance and the control point weights are uniform, then solving for the control point positions is accomplished by linear least squares optimization [4]. This is the approach we take in this paper. One multivariate example is the work of Turner and Crawford [48] who apply NURBS to the modeling of high-dimensional multivariate design spaces. Their application, design space optimization, differs from scientific data analysis, with a relatively small parameter search space, serial modeling, and limited downstream processing of the model.

3 NURBS Tensor Product Background

NURBS, in addition to being able to model complex data with a small number of control points, are piecewise-continuous, differentiable, have local support, and are invariant to affine transformations. To compute the MFA in d dimensions, we use a tensor(d) product of NURBS bases in 1-d. We approximate both field geometry (space-time vertex positions) and science variables (pressure, density, temperature, etc.) in the same representation. The NURBS model is efficiently represented by control data consisting of control points and knots. The control points are reference points that "push and pull" the representation through the linear combination of their products with the basis functions. The knots are scalars in the range $[0.0, 1.0]$ that map partitions of the data to the control points and basis functions.

Table 1: Nomenclature

Symbol	Meaning
x, y, z, t	domain coordinates in space-time
d	number of domain dimensions (e.g., 4)
$\boldsymbol{Q}$	original input data points $\in \mathbb{R}^{d+1}$
u	parameter $\in [0.0, 1.0]$
w	weight associated with a control point, $w > 0$
p	polynomial degree
N	set of basis functions
m	number of input data points
n	number of output control points
$\boldsymbol{P}$	set of n (output) control points $\in \mathbb{R}^{d+1}$
U	set of $n+p+1$ (output) knots $\in [0.0, 1.0]$
$\boldsymbol{R}$	right side of least squares optimization $(N^T N)\boldsymbol{P} = \boldsymbol{R}$
e_{max}	maximum relative error normalized by data extent
e_{rms}	RMS relative error normalized by data extent

Please refer to Table 1 and Figure 1 for the following overview. A curve in (x, y) (Figure 1 left) can be parameterized as a vector-valued function of a single parameter u such that

$$\boldsymbol{C}(u) = \sum_{i=0}^{n-1} N_{i,p}(u)\boldsymbol{P}_i. \tag{1}$$

$N_{i,p}$ are the p-th degree basis functions, and $\boldsymbol{P}$ is the set of n control points; n, the number of output control points, is usually less than m, the number of input data points. The definition extends to higher manifold dimensions (surfaces, volumes, hypervolumes) as a tensor product of multiple parameters $u_1, u_2, ..., u_d$. The right panel of Figure 1 shows a surface, and in general, a d-dimensional hypervolume in $(d+1)$-dimensional space is parameterized as follows:

$$\boldsymbol{V}(u_1, ..., u_d) = \sum_{i_1} ... \sum_{i_d} N_{i_1,p}(u_1) \times ... \times N_{i_d,p}(u_d)\boldsymbol{P}_{i_1,...,i_d}. \tag{2}$$

The basis functions are rational.

$$N_{i,p}(u) = \frac{B_{i,p}(u)w_i}{\sum_{j=0}^{n-1} B_{j,p}(u)w_j}, \tag{3}$$

where B are the nonrational B-spline basis functions and w_i is the weight associated with each control point. In this paper, we set the weights to 1.0; however, we do allow a nonuniform set of weights in our model and account for them when evaluating points from the MFA.

The basis functions are computed using the recurrence formula of Cox [8] and De Boor [10]. The recursive computation of $N_{i,p}(u)$ requires computing $O(p^2)$ nonzero coefficients. The degree p is a small number, usually between 2 and 8 in our experiments. The knots U are computed in $O(n)$ time.

The control points $\boldsymbol{P}$ are found by solving a linear least-squares optimization problem $(N^T N)\boldsymbol{P} = \boldsymbol{R}$, where $\boldsymbol{R}$ is computed from the input data points and basis functions in $O(m+n)$ time [27]. The matrix of basis functions $N^T N$ (in normal form) is n^2 in size, positive definite, and sparse with $2p+1$ nonzero entries along the diagonal [11]. The vector of control points can be solved without pivoting in $O(n^3)$ time. The memory to compute the matrix of basis functions, $N^T N$, is allocated only once for each new dimension and reused; memory size does not grow with the number of dimensions. In future work, we propose representing $N^T N$ in a sparse representation so that its size can be reduced to $O(n(2p+1))$ from the $O(n^2)$ dense representation that we currently use.

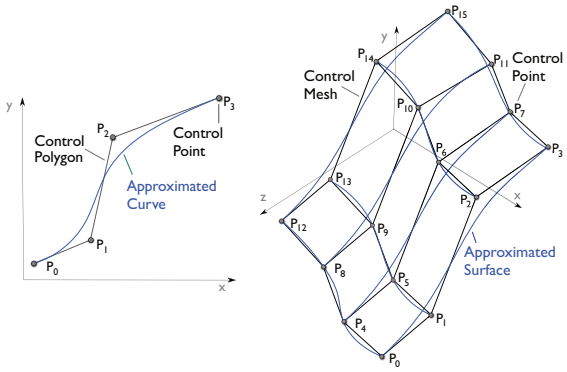

Figure 1: Left: a spline curve. Right: a tensor product of curves in a surface.

4 Modeling the MFA

We continue to use the same nomenclature of Table 1. The user provides the set $\boldsymbol{Q}$ of m input data points, the degree p in each dimension, and the allowable error e_{max}. We define e_{max} to be the maximum relative error, which is the maximum error normalized by the extent of the data range values. Other error metrics are possible with no loss of generality.

4.1 Input Data

We assume the input data points lie in a tensor product of dimension d. Figure 2 shows such a domain for $d = 2$, but our method is not limited to surfaces; this is only for illustration. The tensor product requires the number of points in one dimension of the domain to remain constant while sweeping over the other dimensions. This requirement is illustrated in Figure 2, with one point indicated by a red double circle. The inclusion of this point requires the addition of the other points marked with red circles.

Hierarchical data representations with possibly irregular junctions between tensor product patches have been published to mitigate the tensor product space complexity. Forsey and Bartels [17] used hierarchical refinement to organize patches with different levels of refinement. T-splines are another way to localize the effect of refined regions, so named because they allow T-junctions between tensor product patches. We are studying the work of Sederberg et al. and Bazilevs et al. [47, 3] to determine whether we can organize our tensor product patches into more efficient data models.

The tensor product is topologically a high-dimensional rectangular grid (as opposed to triangular). However, the physical mesh does not have to be strictly regular, as long as there exists a one-to-one mapping of physical coordinates $(x, y, z, ...)$ to $(u, v, w, ...)$ parameters. Typically, complex meshes are first decomposed into rectangular patches, and suitable parameterizations (mappings from physical space to parameter space) are computed for each patch. This is a preprocessing step prior to building the MFA for each patch. The experiments for this paper assume this step has already been completed, and we use regular datasets, assuming the entire domain to be one input patch. The method of Martin et al. [33] converts tetrahedral unstructured volumes into tensor product patches using harmonic functions. In the future, we will investigate this and other methods to preprocess input data.

4.2 Overall Method

Figure 3 illustrates the steps we take to fit the MFA. Beginning with an initial knot distribution for the minimum number of control points, we adaptively add control points and knots until all the evaluated points in each span of knots are within e_{max} of the original points. Knot spans that are out of tolerance are subdivided, and the MFA is recomputed. More details are explained below.

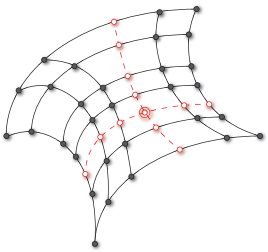

Figure 2: A tensor product of input points. The addition of the circled red point requires the inclusion the other red circles along the dashed curves.

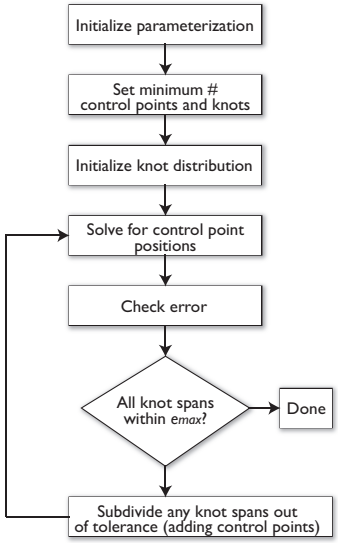

Figure 3: Overall algorithm design.

4.3 Computing Parameterization and Initial Knot Distribution

The first step is to compute parameter values in the range $[0.0, 1.0]$ for all the input data points. In general, the parameterization problem is ill-posed for arbitrary input points [43]. Because the input points in our experiments align with the principal axes, we assigned input point parameters according to the spacing of input points in each of the principal domain dimensions.

Knots are the breakpoints in the parameter space between different basis functions. The number of knots is $n + p + 1$ by definition. In general, the internal knots follow the distribution of the parameters according to the ratio of parameter spans to internal knot spans, $(m-1)/(n-p)$. As with the parameters, we store one vector of knots that is a concatenation of the knots in each dimension, with the total number of knots being the sum of the number of knots in each dimension, not their product.

4.4 Choosing the Degree p

We investigate the relationship between p and error convergence rate with a vanilla input dataset: a 1-d sine function discretized at 1000 points over the domain $[-4\pi, 4\pi]$. We vary the number of control points from 20 to 640 and measure the maximum error in the y direction between the original input points and the corresponding points evaluated from the MFA. Figure 4 compares the error when the MFA is modeled using first through eighth degree basis functions.

On a very smooth function such as $sin(x)$, we can achieve accuracy down to 10^{-14}, essentially the limit of double-precision floating

point accuracy. Moreover given a desired error limit, we can trade off computational complexity (a function of p) and number of control points. In general, how to determine what degree to use for a given input dataset is an open question that we intend to study in our future work. In the present work, we try various degrees for each dataset and select the degree that gives the best combination of small size and low error for that dataset.

We can estimate the order of convergence for different values of p from Figure 4. Since the plot is in log-log scale, the slope of the line is the convergence order. This slope confirms that the convergence order is at least $p+1$, as expected.

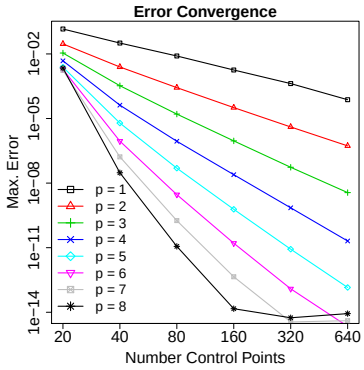

Figure 4: Maximum error of 1-d sine function modeled by an MFA with varying degrees p.

4.5 Separable Multivariate Fitting

Rather than fitting all the dimensions simultaneously, it is less expensive to decompose the dimensions and fit each dimension separately. Assume that the number of domain dimensions is d, and the number of control points is n in each dimension.[1] Solving the least-squares problem in 3-d would take $O(n^{3d})$ time complexity and $O(n^{2d})$ space complexity. By comparison, our separable method takes $O(n^{d+2})$ time complexity and $O(n^2)$ space complexity.

Our method extends that in Piegl and Tiller [42] for 2-d surfaces to any number of dimensions in Algorithm 1 and Figure 5. We approximate the original data in the first dimension to get n control points. In the second dimension, we use the resulting n control points from the first dimension as the input data and fit another set of approximations. This process continues similarly for the following dimensions.

4.6 Adaptive Knot Insertion

We investigated three different adaptive approaches to adding knots: a full-dimensional modeling and evaluation, a one-dimensional modeling and evaluation, and a hybrid full-dimensional modeling and one-dimensional evaluation. The first two approaches produced viable results, while the third method generated an excessive amount of knots and control points without adapting well to different regions of the data. The one-dimensional modeling and evaluation method, while producing a slightly less accurate overall accuracy, runs orders of magnitude faster than the full-dimensional modeling and evaluation, making it our algorithm of choice and described below.

In one-dimensional knot insertion, we treat the model as a set of 1-d curves instead of a unified high-dimensional space. This 1-d approximation is only used to estimate where to add knots. Afterwards, we compute one full multivariate model using Algorithm 1 described in Section 4.5.

[1]Nothing in our method requires the number of control points to be equal across dimensions; this assumption is for illustration only.

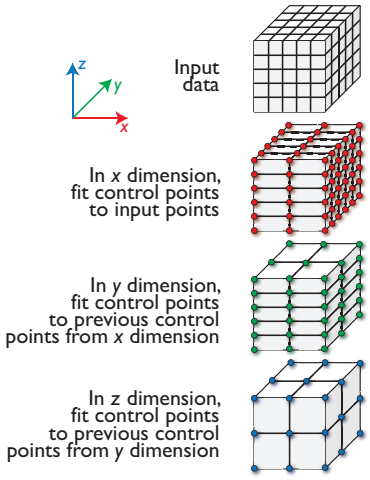

Figure 5: Iterative fitting over separable dimensions reduces the complexity with each dimension because the control points from the previous dimension are the input points to the next dimension.

Algorithm 1 Separable multivariate fitting

```
1:  function MULTIVARIATEFITTING
2:      for all d ∈ domain dimensions do
3:          N^T N ← BasisFuns()
4:          for all curves in current dimension do
5:              if d == 0 then
6:                  Q ← curve points from original domain
7:              else
8:                  Q ← curve points from control points
9:              R ← RHS(Q)
10:             P ← solution of N^T N P = R
11: function BASISFUNS
12:     for all rows in N do
13:         for all knot spans i do
14:             Compute N_{i,p}
15: function RHS
16:     m ← number of input points in current dimension
17:     n ← number of control points in current dimension
18:     for all j ∈ 1, n do
19:         R_j = 0
20:         for all k ∈ 1, m do
21:             R_j = R_j + N_{j,p}(u_k) × Q_k
22:     return R
```

In Algorithm 2, we iterate over the dimensions of the domain in line 8. In each dimension, we loop over curves of input points in line 15. To save time, we first only check 2 curves in the dimension, then 4, then 8, and so on. This stepping through curves is controlled by line 13. We exit this loop when no more knots were found for two consecutive step sizes. As we fit and evaluate the selected curves, we check the knot spans in each evaluated curve and add into a set those spans having at least one domain point that is farther from the curve than e_{max}. From that set of knot spans, we split those (adding a new knot) that would contain at least one input domain point in the new spans. Spans that are too small to contain any input points are discarded.

4.7 Parallel Data Model

Computation of the MFA is parallelized on three levels. First, block parallelism is used to decompose the domain into blocks and execute each block in distributed-memory compute nodes of a supercomputer or computing cluster. We use DIY [36] to accomplish block

Algorithm 2 Adaptive fitting

```
 1: function ADAPTIVE
 2:   while {new_knots} ← NewKnots() ∧{new_knots} ≠ ∅ do
 3:     insert {new_knots} into knot vector
 4:     increment number of control points by ||{new_knots}||
 5:   MultivariateFitting()
 6: function NEWKNOTS
 7:   {new_knots} ← ∅
 8:   for all d ∈ domain dimensions do
 9:     max_nerr ← 0
10:     N^T N ← BasisFuns()
11:     ncurves ← number curves in d direction
12:     S ← spans to be split in d direction ← ∅
13:     for all step ∈ [ncurves/2, 1] by step ← step/2 do
14:       if ncurves/step < max_ncurves then
15:         for all curves C ∈ dimension d by step do
16:           Q ← curve points from original domain
17:           R ← RHS(Q)
18:           P ← solution of N^T N P = R
19:           for all knot spans s ∈ C do
20:             for all input points dp ∈ s do
21:               cp ← evaluated point at dp
22:               if |dp − cp| > e_max then
23:                 insert s in S
24:                 break
25:
26:           num_err ← number of domain points in C farther than
              e_max from C
27:           if num_err > max_nerr then
28:             max_nerr ← num_err
29:       if max_nerr is unchanged then
30:         break
31:     for all spans s ∈ S do
32:       if s can be split then
33:         bisect s and insert midpoint knot in {new_knots}
34:   return {new_knots}
```

parallelism. Within a block, task parallelism is utilized to fit curves in thread-level tasks. Currently we use TBB [41], but in principle other task-parallel programming models can be used. The tasks assign threads to parallelize the MFA computation over different curves in the same dimension. Those computations are independent, so they parallelize well. Within a curve, linear algebra operations, e.g., to invert matrices, are vectorized. Currently we use Eigen [20] for this purpose.

Each block of the parallel data model is a tensor product defined by n control points and $n+p+1$ knots in each dimension. Control points are in the same coordinate system as original data points, meaning that the same spatiotemporal domain decomposition used for input data can be reused for the MFA, minimizing data movement. The basis functions are not stored and are recomputed as needed. The data reduction is primarily governed by the ratio between the number of input data points to output control points. Assuming the same reduction in each dimension, the total reduction is exponential in the number of dimensions.

The output MFA is stored in a binary file in DIY format, which is read and written in parallel using MPI-I/O. We also wrote a serial utility to convert the file to VTK format so that the results are compatible with visualization and analysis tools derived from VTK such as ParaView and VisIt. (The images of the MFA in Section 6 are generated by ParaView.) In our future work, we plan to implement a reader for those tools to read the DIY file in parallel, as well as modifying VTK filters to operate directly on the MFA. One of the reasons we chose a NURBS geometric basis for the MFA is because of its compatibility with such downstream tasks.

5 USING THE MFA

The MFA is designed to be used for subsequent data analysis and visualization, distinguishing it from other basis representations such as wavelets, cosine transformations, and compression algorithms that require the inverse transform to first be applied. Several operations are possible directly from the MFA without reverting to the original discrete data model, and moreover they are possible in the full order and accuracy of the model, without linear interpolation or finite difference estimation. While the model itself is an approximation to the original data, subsequent applications of the MFA result in no further approximation. Once the model is computed according to Section 4, one can evaluate the model at any point in the domain, not just at the original input points, and differentiate the model up to the $p-th$ derivative in any combination of partial derivatives in the domain dimensions. Affine transformations can be applied directly to the control points in Equations 1 - 3. In the future, we will also investigate statistical and machine learning algorithms using the MFA, such as clustering and principal components analysis.

Today, the MFA is not a drop-in replacement for discrete datasets in analysis and visualization tools, and algorithms and tools would need to be modified to ingest the MFA data model. However, we argue that such effort is justified if it allows data analytics and visualization to keep pace with exascale computation. Future work will be needed to demonstrate some of those downstream algorithms and profile their performance. To date, we can evaluate points and derivatives anywhere in the domain directly from the MFA, which is fundamental to many subsequent algorithms. The following sections explain multivariate evaluation and high-order differentiation.

5.1 Multivariate Evaluation

Figure 6 shows an example of evaluating a 2-d point with $p=2$ in both dimensions. The number of basis functions and control points multiplied in each dimension is $p+1$; hence, 9 control points are shown in image (a). First, each curve in the first (horizontal) dimension is collapsed into a single point by the matrix multiplication of basis functions and control points. This step is shown in images (b) and (c). Next, the 3 resulting points form a curve in the vertical direction, which is collapsed in (d) through multiplication of basis functions to become the resulting evaluated point in (e).

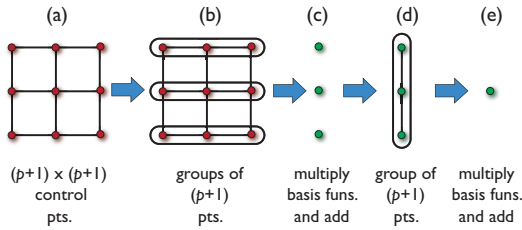

Figure 6: Steps to evaluate a point from the MFA.

We extend this method to any number of dimensions in Algorithm 3. In order to handle arbitrary dimensionality, we convert what logically is a nested loop structure of $p+1$ iterations in each dimension into a flat iteration space of the total product of the nested iterations. There is some additional bookkeeping not shown in the pseudocode that determines which basis functions to multiply by which control point, based on the iteration number in the flattened iteration space. For example, in lines 8 and 11 of Algorithm 3, the notation $control\_pt_{d,i}$ refers to a particular control point in the d-th dimension based on the i-th iteration.

Algorithm 3 has two main steps. The loop in lines 3-6 computes the basis functions for the parameter value of the point to be evaluated, and the loop in lines 7-12 multiplies $p+1$ nonzero coefficients

Algorithm 3 Multivariate evaluation

```
1:  function EVALUATE
2:      tot_iterations ← 1
3:      for all d ∈ domain dimensions do
4:          N_d ← BasisFuns()
5:          tot_iterations ← tot_iterations ∗ (p_d + 1)
6:          temp_d ← 0.0
7:      for all i ∈ tot_iterations do
8:          temp_0 ← temp_0 + N_{0,i} × control_pt_{0,i}
9:          for all d ∈ domain dimensions do
10:             if i mod p_d + 1 then
11:                 temp_{d+1} ← temp_{d+1} + N_{d+1,i} × control_pt_{d+1,i}
12:                 temp_d ← 0.0
```

by control points. The numerical complexity of the two steps is $O(p^2)$ and $O(p^d)$, respectively.

5.2 High-order Differentiation

Differentiation, including high-order derivatives, is a staple of scientific data analysis and visualization. Gradient fields, velocities, Jacobian matrices, edge detection, topological segmentation, and uncertainty quantification all require first derivatives. Second derivatives are used for computing curvature, acceleration, and Hessian matrices. For example, ridge and valley features are defined in terms of gradients and eigenvalues of the Hessian [15]. Applying lighting and shading to ridge features requires their normal, or the third derivative of the original model.

The first p derivatives of a p-degree basis function are computed with a similar recurrence formula as the basis function values. For example, the $p = 3$ basis functions and their first through third derivatives are shown in Figure 7. With derivatives of basis functions in hand, MFA derivatives are computed by substituting N with N' in Equation 2 to yield Equation 4.

$$\frac{\partial^{k_1+\ldots+k_d}}{\partial^{k_1}u_1,\ldots,\partial^{k_d}u_d}\boldsymbol{V}(u_1,\ldots,u_d) = \sum_{i_1}\ldots\sum_{i_d} N^{(k_1)}_{i_1,p}(u_1)\times\ldots\times N^{(k_d)}_{i_d,p}(u_d)\boldsymbol{P}_{i_1,\ldots,i_d}. \quad (4)$$

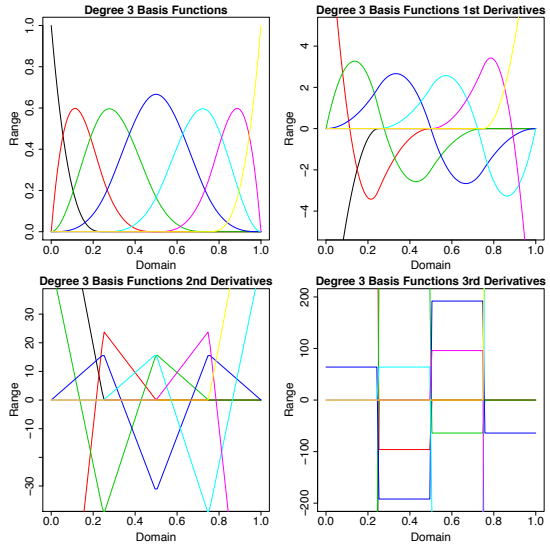

Figure 7: Degree-3 basis functions and derivatives.

Basis functions and their derivatives are not stored in the data model and are recomputed as needed. Because derivatives of the MFA use the same knots and control points as the original MFA, we compute the MFA once using original values and store the model in a file. Derivatives are computed in postprocessing from the stored model. This means that we only need to store one MFA irrespective of how many times we want to evaluate or differentiate it later. The only requirement is to fit the original MFA with high enough degree p such that any later derivatives are less than or equal to p in degree. This includes the sum of partial derivatives in Equation 4, i.e., $k_1 + \ldots + k_d <= p$.

Figure 8 demonstrates the use of our MFA to compute the derivative of the sinc function described in Section 6.1. We modeled an original set of 100 input points with 20 control points using $p = 5$. The evaluated values of the MFA appear in the taller black curve of Figure 8. Subsequently, we differentiated the MFA where the y coordinate is the first derivative of the MFA. This is the shorter red curve. It is easy to visually confirm that the slope of the black curve corresponds to the value of the red curve.

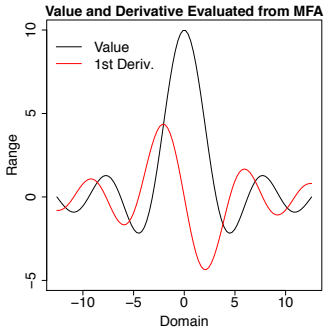

Figure 8: 1-d sinc function in black (taller curve) and its first derivative in red (shorter curve) illustrate point evaluation and differentiation from an existing MFA.

6 Experimental Results

We performed two sets of experiments. In the first, we test the adaptive fitting by automatically adding knots and control points until a desired error threshold is reached. In the second, we test parallel scalability for a fixed number of control points.

The first set of experiments was conducted on a laptop with an Intel 2.7 GHz Core i7 CPU with 4 hardware cores and 8 threads. It contains 16 GB of DDR3 RAM and a 500 GB SSD. Our code was compiled with LLVM version 9.0.0 (Clang version 900.0.38) with -O3 optimization.

The second set of experiments was conducted on the Theta supercomputer at the Argonne Leadership Computing Facility of Argonne National Laboratory. Theta is a Cray XC40 machine with 4,392 nodes. Each node has one Intel Xeon Phi Knights Landing 64-core CPU, 16 GB high-bandwidth MCDRAM, 192 GB DDR4 RAM, and 128 GB of SSD storage. Theta is interconnected with a Cray Aries 3-level Dragonfly network, and the machine has a peak aggregate compute rate of 11.7 PFLOPs. We used the Intel compiler with -O3 optimization.

6.1 Synthetic Data

Sinc is a synthetic dataset of the cardinal sine function, $y = sin(x)/x$, that we can generate in any dimensionality and resolution. In order to increase the range and slope of the data, we scaled the sinc function by a factor of 10. The 1-d sinc function is $f(x) = 10sin(x)/x$. In 2-d, $f(x,y) = 10sinc(x)sinc(y)$, and so forth for higher dimensions. Figure 9 shows a 2-d sinc function. The sinc function was chosen for its smoothness; because of its high degree of continuity, the MFA

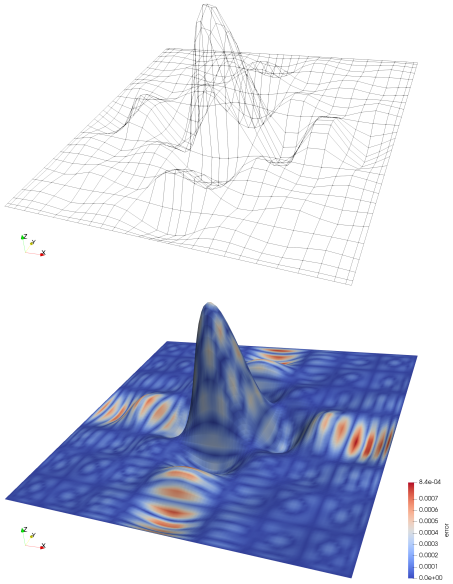

Figure 9: Top: Adaptive grid of 784 control points for 2-d sinc function with desired $e_{max} = 1 \times 10^{-4}$. Bottom: Original input data colored by error compared with reconstructed data.

is able to model such data efficiently using high degree (we used $p = 5$). In essence, this experiment shows the best-case behavior.

Table 2 shows our adaptive fitting algorithm (Algorithm 2) on a 4-d sinc function. We varied e_{max} and measured the output number of control points, resulting error, number of iterations, and modeling time. Because our algorithm estimates knot insertion in a low-dimensional space, the actual value of e_{max} does not always match the desired value, but the actual error is always within the same order of magnitude or better. The tradeoff is speed: adaptively modeling 100 million data points can be achieved in approximately one minute on a single-node machine. The time is related to the number of refinement iterations needed to achieve the desired accuracy and generally increases with lower error bound, while the number of control points generally increases. Although we are using normalized maximum error as our error control, Table 2 also reports the normalized RMS error for reference. The RMS error is one to two orders of magnitude less than maximum error.

Table 2: 4-d Sinc Dataset w/ 10^8 Input Points

Desired e_{max}	Output Ctrl Pts	Cmpr Fctr	Actual e_{max}	Actual e_{rms}	Iters	Time (s)
10^{-2}	2.9×10^4	3499	3.9×10^{-2}	5.7×10^{-4}	4	35.0
10^{-3}	1.9×10^5	514	5.2×10^{-4}	6.2×10^{-6}	5	37.9
10^{-4}	5.3×10^5	188	8.5×10^{-5}	2.7×10^{-6}	8	87.9
10^{-5}	1.9×10^6	53	3.9×10^{-6}	4.5×10^{-8}	6	60.0

6.2 Scientific Data

In the following experiments, we continue to use the adaptive fitting method described in Section 4.6 and Algorithm 2. Unlike the sinc data, these datasets are ill-conditioned, with sharp edges and high-frequency details, and are representative of actual data one would encounter in scientific experiments or simulations.

6.2.1 3-d Combustion Datset

S3D is a turbulent combustion data set generated by an S3D simulation [7] of fuel jet combustion in the presence of an external cross-flow [19, 18, 40, 39]. The domain is 3-d (x, y, z) $(704 \times 540 \times 550)$, and the range variable $f(x, y, z)$ is the magnitude of the 3-d velocity. We slice this dataset to produce 1-d, and 2-d cross-sections, in addition to using it in its original 3-d form.[2] We used $p = 3$ for modeling the S3D dataset.

Figure 10 shows the progress of the adaptive fitting algorithm as the error rate converges to the user-specified limit. This test uses the 2-d S3D dataset, with $p = 3$, and $e_{max} = 10^{-2}$. The starting error after the first iteration is 8.3×10^{-1}, and the ending error after 11 iterations is 1.5×10^{-1}. The progress is nonlinear; the first few iterations reduce the error slowly, but the middle iterations reduce the error steadily until the error reaches a minimum at iteration 9 and does not reduce further. We continue to work making the adaptive algorithm more accurate as well as accelerating its convergence.

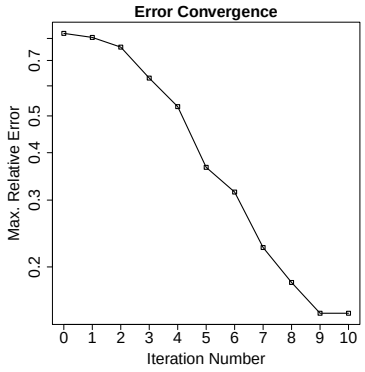

Figure 10: Convergence of maximum relative error to the desired threshold as a function of the number of iterations of the adaptive fitting algorithm. 2-d S3D dataset with $e_{max} = 10^{-2}$. Plotted in semilog scale.

For the same experiment, the top image in Figure 11 shows the resulting control points. Notice how the adaptive algorithm clustered control points in the turbulent regions of high velocity gradient. The relatively flat regions in the dataset are modeled with a more coarse resolution of control points.

Table 3 shows adaptive fitting of the 3-d S3D dataset. We varied e_{max} and measured the output number of control points, resulting error, number of iterations, and modeling time. In this case, our adaptive algorithm does not achieve the desired e_{max} error. The RMS error is two orders of magnitude better, indicating that the areas of worst fit are localized to a small region, as evident in Figure 11.

Setting a smaller error bound does not always require more iterations and longer time. As we see in this result and in the next two datasets, sometimes a tighter error tolerance can be attained by adding more knots in earlier iterations, making the adaptive algorithm converge sooner.

Table 3: 3-d S3D Dataset w/ 2.1×10^8 Input Points

Desired e_{max}	Output Ctrl Pts	Cmpr Fctr	Actual e_{max}	Actual e_{rms}	Iters	Time (s)
10^{-1}	4.2×10^4	5025	4.0×10^{-1}	7.5×10^{-3}	18	479
10^{-2}	1.8×10^6	117	2.5×10^{-1}	2.1×10^{-3}	16	197

6.2.2 3-d Thermal Hydraulics Dataset

The next case study is a 3-d vector field representing the numerical results of a large-eddy simulation of Navier-Stokes equations for the

[2] Our use of the terms 1-d, 2-d, or 3-d is the number of dimensions in the domain only, not the domain plus the range, which would include the function value or science variables.

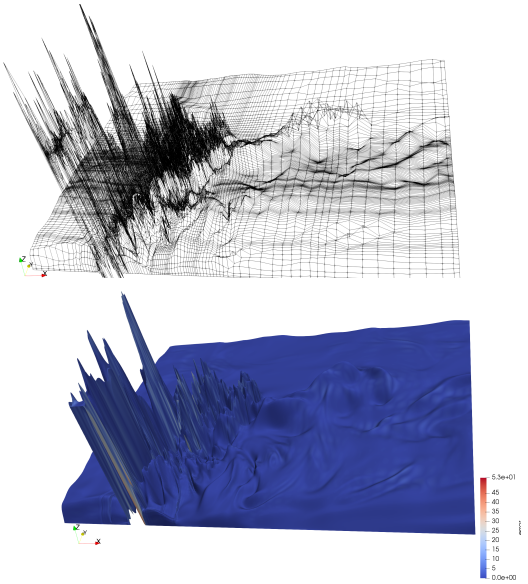

Figure 11: Top: Adaptive grid of 18,476 control points for 2-d S3D data with desired $e_{max} = 1 \times 10^{-2}$. Bottom: Original input data colored by actual e_{max}.

MAX experiment [35] conducted at Argonne National Laboratory in 2010. The problem is representative of turbulent mixing and thermal striping that occurs in the upper plenum of liquid sodium fast reactors. The dataset is courtesy of Aleksandr Obabko and Paul Fischer of Argonne National Laboratory and is generated by the Nek5000 solver. The data have been resampled from their original topology onto a $200 \times 200 \times 200$ regular grid, and the magnitude of the velocity vector is associated with each 3-d domain point.

Table 4 shows adaptive fitting with $p = 3$. Accuracy is similar to the S3D data, but the compression factor is minimal. Figure 12 shows a volume rendering of reconstructed data from Nek5000.

Table 4: 3-d Nek5000 Dataset w/ 8.0×10^6 Input Points

Desired e_{max}	Output Ctrl Pts	Cmpr Fctr	Actual e_{max}	Actual e_{rms}	Iters	Time (s)
10^{-1}	1.1×10^6	7.6	2.3×10^{-1}	1.7×10^{-2}	12	49.3
10^{-2}	8.0×10^6	1.0	8.8×10^{-2}	7.6×10^{-4}	8	22.7

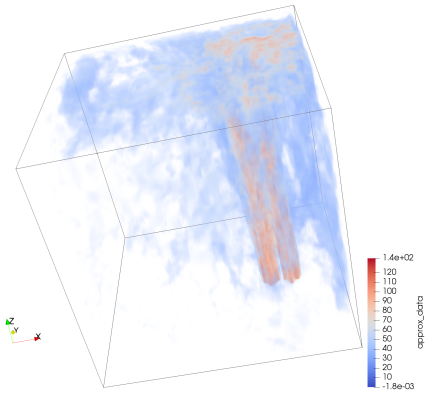

Figure 12: Volume rendering of data evaluated from the MFA modeled with desired $e_{max} = 10^{-2}$.

6.2.3 2-d Climate Dataset

The Community Earth System Model (CESM) is global climate data of the Earth's oceans, atmosphere, land, and sea ice. The dataset we modeled is the FLDSC (Clearsky downwelling longwave flux at surface) variable of the Community Atmosphere Model (CAM) developed at the National Center for Atmospheric Research (NCAR) [37]. Our dataset is an 1800×3600 2-d domain with one value of the FLDSC science variable at each grid point.

Table 5 shows adaptive fitting with $p = 3$. Figure 13 shows the superposition of CESM control points, input data points, and evaluated points.

Table 5: 2-d CESM Dataset w/ 6.5×10^6 Input Points

Desired e_{max}	Output Ctrl Pts	Cmpr Fctr	Actual e_{max}	Actual e_{rms}	Iters	Time (s)
10^{-2}	9.0×10^4	71	3.3×10^{-1}	6.1×10^{-2}	20	1806
10^{-3}	1.7×10^6	3.8	3.0×10^{-2}	4.8×10^{-4}	12	1653

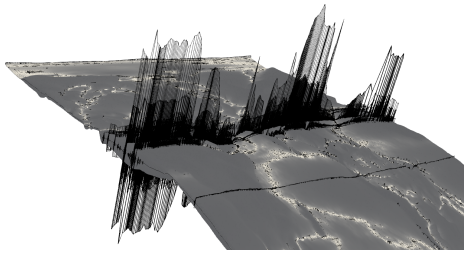

Figure 13: CESM input dataset. Height is velocity magnitude at horizontal grid locations. Adaptive control points, input data points, and evaluated points from the MFA are superimposed.

6.3 Discussion

Although we report compression factor in our results, we do not attempt to compete head-to-head with lossy compression algorithms such as those listed in the related work. In fact, readers familiar with that literature will immediately see that our results are worse; the reason being that unlike wavelets or other lossy compressors, the MFA is a transformed data model that retains geometric properties and enables analytical operations directly without decompressing. Section V shows some of these operations by evaluating points and derivatives (up to degree p) analytically from the model at any location. The tradeoff in this transformation is that the rate distortion curve (the relationship between data size and error) for nonsmooth, noisy data such as Nek5000, S3D, and CESM is worse for the MFA than for lossy compressors that encode every data point along with the residual error. Rather than adjusting the number of bits used to encode all the data points, we attempt to achieve the user-specified error rate by automatically selecting the number and location of control points with our adaptive knot insertion algorithm.

6.4 Parallel Scalability

We return to the synthetic sinc dataset. Solely for the purpose of illustration, Figure 14 shows a 2-d surface decomposed into 36 blocks, with each block modeled in parallel in a separate MFA. In the next experiment, we used a 3-d volume rather than a 2-d surface, with a large number of input points, decomposed into many blocks, executing over thousands of MPI processes. The experiment was conducted on the Theta supercomputer at the Argonne Leadership Computing Facility.

We conducted both strong and weak scalability tests. In these tests, we fixed the number of control points at $1/5$ the number of

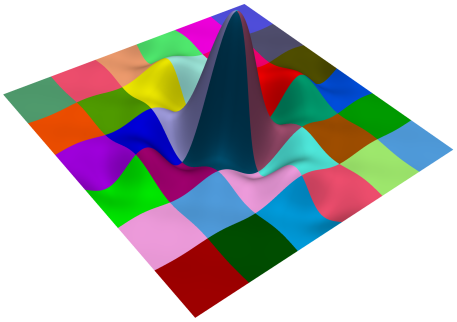

Figure 14: 2-d sinc synthetic dataset partitioned into 36 blocks and modeled in parallel, blocks colored by process ID.

input points and did not adapt control points with respect to a user error bound. For strong scaling, the total number of input points and total number of output control points remains constant as MPI processes are added. We used 1000^3 input points and 200^3 output control points for a compression factor of $5^3 = 125$. In weak scaling, both the number of input points and the number of control points per MPI process remain constant. We used 200^3 input points per block and 40^3 output control points per block in the weak scaling test. The largest test in the weak scaling regime had over 32 billion input points.

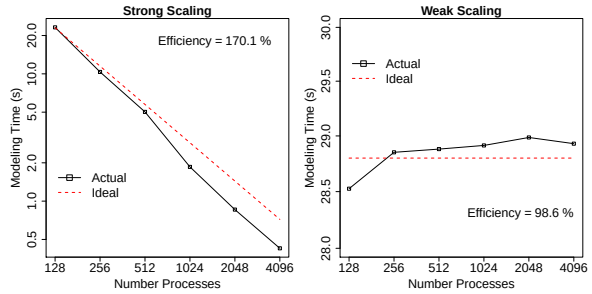

Figure 15: Strong and weak scaling.

Thus far, both the strong and weak scaling results in Figure 15 are nearly 100% efficient. This is because the MFA has local support, and its geometric properties mimic the locality of the original data; i.e., no communication is needed. There is a section of superlinear speedup in the strong scaling curve between 512 and 1024 processes that we are investigating. The results so far show that the problem is parallelizable, and we are optimistic that there is headroom for some communication overhead and load imbalance while maintaining good scalability, although this is yet to be proved.

We are currently studying to what extent to overlap information between blocks in the parallel decomposition. With no overlap (the current result), all errors, including at block boundaries, are within e_{max}, but continuity is not enforced across block boundaries. This means that the multiblock parallel and single-block serial results are within e_{max} of each other, but not identical. This may be acceptable, but if tighter continuity is required across blocks, we are working to determine how much information to exchange across blocks and how to constrain the fitting inside a block by the neighboring blocks. We also need to measure parallel performance over a broad range of datasets—real and synthetic—with different load balance characteristics.

7 Conclusion

In this paper, we developed the basics of modeling the MFA in high-dimensional space, with adaptive refinement, in parallel. We further implemented high-dimensional functions to evaluate points and high-order derivatives anywhere in the domain. This capability will allow comparing two models that originated from different discretizations. Applications include not only comparative analysis but also multiphysics model coupling. Another application is nonuniform spatial smoothing by increasing the degree of fit and/or reducing the number of knots and control points.

NURBS models have some well-known limitations, and we are pursuing multiple research directions to overcome them. One problem is accurate modeling of noisy, high-frequency data. Section 6.2 showed that accuracy and compression for scientific data are worse than state-of-the-art floating-point lossy compression algorithms. Moreover, the largest errors occurred in regions of steepest gradient in the input data. To that end, we are experimenting with various approaches to assigning nonuniform weights to control points using both linear and nonlinear optimization methods. We also continue to investigate different algorithms for adaptively adding control points—including increasing knot multiplicity at adaptive segment boundaries—and to study the resulting local and global error.

Our research in block parallelization is ongoing. We intend to allow various degrees of user-specified continuity at block boundaries, and we are actively studying the problem from two directions: imposing additional constraints during the modeling in order to satisfy the desired level of interblock continuity, and developing blending methods that satisfy continuity during the evaluation of points from independently fitted blocks. Another consideration is selecting an appropriate domain decomposition to preserve data features and/or balance computational load. In our parallel data model, the degree p can vary between blocks, motivating automatic selection of p.

Another limitation of tensor products is their space complexity with increasing dimensionality. Because tensor products require the number of control points in one dimension to remain constant when sweeping over the other dimensions, inserting a knot in one dimension results in hyperplanes (in the parameter space) of additional control points. In addition to investigating T-splines and other data organization methods to allay this problem, we are also working to fit separate NURBS models for the field geometry and for each science variable. In addition to lowering the dimensionality of the models, data size can be further reduced because in many cases the field geometry requires fewer control points than the science variables.

Continued application to more datasets is another ongoing activity. We will investigate unstructured, adaptive refinement, and particle data, both simulation and experimental. We will include more error metrics such as peak signal-to-noise ratio in addition to maximum and RMS error. With improved adaptive fitting algorithms and nonuniform weights, we will be able to compare head-to-head with compression algorithms from the literature. In terms of downstream uses of the MFA, we will modify visualization algorithms to directly ingest the model, including writing custom plugins for ParaView or VisIt to directly evaluate the MFA representation. Furthermore, we will experiment with the MFA for machine learning tasks such as computing statistics, clustering, or principal components analysis.

Acknowledgments

This work is supported by Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357, program manager Laura Biven. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

REFERENCES

[1] W. Austin, G. Ballard, and T. G. Kolda. Parallel Tensor Compression for Large-Scale Scientific Data. *arXiv preprint arXiv:1510.06689*, 2015.

[2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., *Modern Software Tools in Scientific Computing*, pp. 163–202. Birkhäuser Press, 1997.

[3] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric Analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010.

[4] A. Bjorck. *Numerical Methods for Least Squares Problems*. Siam, 1996.

[5] B. Boashash. *Time-frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic Press, 2015.

[6] C.-M. Chen, A. Biswas, and H.-W. Shen. Uncertainty Modeling and Error Reduction for Pathline Computation in Time-Varying Flow Fields. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific*, pp. 215–222. IEEE, 2015.

[7] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo. Terascale Direct Numerical Simulations of Turbulent Combustion using S3D. *Comput. Sci. Disc.*, 2:015001, 2009.

[8] M. G. Cox. The Numerical Evaluation of B-splines. *IMA Journal of Applied Mathematics*, 10(2):134–149, 1972.

[9] L. Dalcin, N. Collier, P. Vignal, A. Crtes, and V. Calo. PetIGA: A Framework for High-performance Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering*, 308:151–181, 2016. doi: 10.1016/j.cma.2016.05.011

[10] C. De Boor. On Calculating with B-splines. *Journal of Approximation theory*, 6(1):50–62, 1972.

[11] C. De Boor. A Practical Guide to Splines. *Mathematics of Computation*, 1978.

[12] C. De Boor. A Practical Guide to Splines (revised ed.). *New York*, 2001.

[13] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-order Methods for Incompressible Fluid Flow*. Aug. 2002.

[14] S. Di and F. Cappello. Fast Error-bounded Lossy HPC Data Compression with SZ. pp. 730–739, 2016.

[15] D. Eberly. *Ridges in Image and Data Analysis*, vol. 7. Springer Science & Business Media, 2012.

[16] F. Ferraty and P. Vieu. *Nonparametric Functional Data Analysis: Theory and Practice*. Springer Science & Business Media, 2006.

[17] D. R. Forsey and R. H. Bartels. Hierarchical B-spline Refinement. *ACM Siggraph Computer Graphics*, 22(4):205–212, 1988.

[18] T. F. Fric and A. Roshko. Vortical Structure in the Wake of a Transverse Jet. *Journal of Fluid Mechanics*, 279:1–47, 1994.

[19] R. W. Grout, A. Gruber, C. Yoo, and J. Chen. Direct Numerical Simulation of Flame Stabilization Downstream of a Transverse Fuel Jet in Cross-flow. *Proceedings of the Combustion Institute*, 33:1629–1637, 2011.

[20] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[21] C. E. Heil and D. F. Walnut. Continuous and Discrete Wavelet Transforms. *SIAM review*, 31(4):628–666, 1989.

[22] J. Hua, Y. He, and H. Qin. Multiresolution Heterogeneous Solid Modeling and Visualization Using Trivariate Simplex Splines. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, pp. 47–58. Eurographics Association, 2004.

[23] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4135–4195, 2005.

[24] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. Out-of-Core Compression and Decompression of Large N-Dimensional Scalar Fields. In *Computer Graphics Forum*, vol. 22, pp. 343–348. Wiley Online Library, 2003.

[25] M. H. Jansen and P. J. Oonincx. *Second Generation Wavelets and Applications*. Springer Science & Business Media, 2005.

[26] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data. In *Euro-Par 2011 Parallel Processing*, pp. 366–379. Springer, 2011.

[27] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting*. Academic Press, 1986.

[28] D. Laney, S. Langer, C. Weber, P. Lindstrom, and A. Wegener. Assessing the Effects of Data Compression in Simulations using Physically Motivated Metrics. In *2013 SC-International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12. IEEE, 2013.

[29] H. Lin, T. Maekawa, and C. Deng. Survey on Geometric Iterative Methods and their Applications. *Computer-Aided Design*, 95:40–51, 2018.

[30] P. Lindstrom. Fixed-rate Compressed Floating-point Arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[31] P. Lindstrom and M. Isenburg. Fast and Efficient Compression of Floating-point Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

[32] Z. Majdisova and V. Skala. Radial Basis Function Approximations: Comparison and Applications. *Applied Mathematical Modelling*, 51:728–743, 2017.

[33] T. Martin, E. Cohen, and M. Kirby. Volumetric Parameterization and Trivariate B-spline Fitting using Harmonic Functions. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, pp. 269–280. ACM, 2008.

[34] W. Martin and E. Cohen. Representation and Extraction of Volumetric Attributes Using Trivariate Splines: A Mathematical Framework. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pp. 234–240. ACM, 2001.

[35] E. Merzari, W. Pointer, A. Obabko, and P. Fischer. On the Numerical Simulation of Thermal Striping in the Upper Plenum of a Fast Reactor. In *Proceedings of ICAPP 2010*. San Diego, CA, 2010.

[36] D. Morozov and T. Peterka. DIY2: Data-parallel Out-of-core Library. In *Proceedings of the 2016 IEEE Large Data Analysis and Visualization Symposium LDAV'16*. Baltimore, MD, 2016.

[37] R. B. Neale, C.-C. Chen, A. Gettelman, P. H. Lauritzen, S. Park, D. L. Williamson, A. J. Conley, R. Garcia, D. Kinnison, J.-F. Lamarque, et al. Description of the NCAR Community Atmosphere Model (CAM 5.0). Technical report, 2010.

[38] S. Park and K. Lee. High-dimensional Trivariate NURBS Representation for Analyzing and Visualizing Fluid Flow Data. *Computers & Graphics*, 21(4):473–482, 1997.

[39] T. Peterka, R. Ross, W. Kendall, A. Gyulassy, V. Pascucci, H.-W. Shen, T.-Y. Lee, and A. Chaudhuri. Scalable Parallel Building Blocks for Custom Data Analysis. In *Proceedings of the 2011 IEEE Large Data Analysis and Visualization Symposium LDAV'11*. Providence, RI, 2011.

[40] T. Peterka, R. Ross, B. Nouanesengsy, T.-Y. Lee, H.-W. Shen, W. Kendall, and J. Huang. A Study of Parallel Particle Tracing for Steady-state and Time-varying Flow Fields. In *Proceedings of IPDPS 11*. Anchorage AK, 2011.

[41] C. Pheatt. Intel® Threading Building Blocks. *Journal of Computing Sciences in Colleges*, 23(4):298–298, 2008.

[42] L. Piegl and W. Tiller. *The NURBS Book*. 1997.

[43] L. A. Piegl and W. Tiller. Parametrization for Surface Fitting in Reverse Engineering. *Computer-Aided Design*, 33(8):593–603, 2001.

[44] J. O. Ramsay, G. Hooker, and S. Graves. *Functional Data Analysis with R and MATLAB*. Springer Science & Business Media, 2009.

[45] J. O. Ramsay and B. W. Silverman. *Applied Functional Data Analysis: Methods and Case Studies*, vol. 77. Citeseer, 2002.

[46] A. Raviv and G. Elber. Interactive Direct Rendering of Trivariate B-spline Scalar Functions. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):109–119, 2001.

[47] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM transactions on graphics (TOG)*, 22(3):477–484, 2003.

[48] C. J. Turner and R. H. Crawford. N-dimensional Nonuniform Rational B-splines for Metamodeling. *Journal of Computing and Information Science in Engineering*, 9(3):031002, 2009.